In-Place Augmented Reality

Nate Hagbi*
Computer Science Department
Ben-Gurion University
Israel

Oriel Bergig*
Computer Science Department
Ben-Gurion University
Israel

Jihad El-Sana*
Computer Science Department
Ben-Gurion University
Israel

Klara Kedem*
Computer Science Department
Ben-Gurion University
Israel

Mark Billinghurst†
The HIT Lab NZ
University of Canterbury
New Zealand

ABSTRACT

In this paper we present a new vision-based approach for transmitting virtual models for Augmented Reality (AR). A two dimensional representation of the virtual models is embedded in a printed image. We apply image-processing techniques to interpret the printed image and extract the virtual models, which are then overlaid back on the printed image. The main advantages of our approach are: (1) the image of the embedded virtual models and their behaviors are understandable to a human without using an AR system, and (2) no database or network communication is required to retrieve the models. The latter is useful in scenarios with large numbers of users. We implemented an AR system that demonstrates the feasibility of our approach. Applications in education, advertisement, gaming, and other domains can benefit from our approach, since content providers need only to publish the printed content and all virtual information arrives with it.

KEYWORDS: Augmented Reality content, content transmission, model embedding, dual perception encoding.

INDEX TERMS: H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities; I.4.0 [Image Processing and Computer Vision]: Image processing software.

1 INTRODUCTION

Augmented Reality (AR) enhances user perception by supplementing the real world with virtual content. The virtual content is commonly stored in a local or remote library and is fetched by the AR system for rendering. In this paper we present an alternative approach where the content is encoded in the real world itself.

It is a complicated task to update libraries which are stored localy on multiple devices running an AR application. Remote databases, on the other hand, are easy to update, but require a network communication infrastructure. In addition, since a large number of content providers may be involved in creating the content, locating it by users may become a complicated process.

Managing content poses difficulties for designing AR systems for massive numbers of users. In cases of large simultaneous demand for the same AR content a significant load is placed on the content servers. Another difficulty is that printed material has to be published, in addition to virtual content being distributed electronically. For example, advertisers may have to publish their advertisment in the newspaper in addition to making the virtual content available over a cellular network. The need to create, store, and transmit virtual models thus poses challenges for AR applications. A convenient way to transmit virtual models without having to maintain a content library would lay the foundations for a new type of AR applications.

This paper introduces the idea of vision-based transmission of AR models, which combines the advantages of local and remote content libraries. The transmission is achieved by embedding virtual model content in images. The embedded models are then captured and extracted by the AR system and the extracted models are used for virtual augmentation of the real images. We name this novel scheme In-Place Augmented Reality (IPAR), because the AR content is embedded in the same place (e.g. on printed-paper) in which it is viewed. Our approach aims to provide *dual perception*. Namely, we encode virtual models and their behaviors, such that the embedded representation reflects the underlying content also without the use of an AR system.

The main advantage of our approach is that content delivery is made easier. Since the AR tracking information and the AR content are contained in one printed image, there is no need to ask the user to send an identification code to the content provider. At the same time, content providers do not have to distribute their content electronically over a communication network. Another advantage of our approach is that a content library and the infrastructure required to support it, such as a database and communication network, are not required. In some cases, this can meaningfully simplify the design and maintenance of the system. The dual perception property in our approach is another advantage. Since the printed image reflects the augmented content, interaction with the printed image becomes more natural.

However, In-Place Augmented Reality inherits some of the limitations common in vision-based methods. The imaging device capabilities directly affect the quality of extracted models. In addition, the basic approach is limited to applications where content does not have spatial context. Furthermore, supporting complex 3D models is not trivial and in this paper we only address 2.5D models by embedding elevation maps.

The rest of this paper is structured as follows. In the next section we describe related work. Section 3 describes the concept and details regarding model embedding, acquisition, and extraction. Section 4 discusses implementation and performance issues and Section 5 describes several IPAR applications. Section 6 concludes and describes future work.

2 RELATED WORK

The general concept of relying on vision for content retrieval is not new. Barcodes are widely used as a vision-based mechanism for identification [1]. In AR systems visual markers are commonly used. These typically serve two purposes: (1) to identify real objects, and (2) to calculate the camera pose relative to the real objects. For example, ARToolKit [2] uses square fiducial markers, where the pattern inside the square is used to decide

* {natios, bergig, el-sana, klara}@cs.bgu.ac.il
† mark.billinghurst@hitlabnz.org

Figure 1. Augmentation of a Saint Moritz map. (a) The embedded image. (b) The augmented image.

which model should be augmented and the square is used to calculate the camera pose. ARTag [3], MXRToolKit [4] and stbTracker [5] are other examples of AR libraries that use a similar two step technique.

Marker patterns have also been utilized for conveying richer content. In AR Storyboard [6] patterns are used to compose stories, where properties are assigned to characters by combining the appropriate patterns. In [7] an AR system makes use of barcodes to determine the context, which is used by a remote database server to deliver context-aware content. Some other AR environments combining barcodes are described in [8]. In the AR PlayStation game 'The Eye of Judgment' [9], cards are used to identify models and actions that can be performed. The appearance of the cards reflects the content they identify.

The main distinguishing factor of our work is that the AR content itself is retrieved from natural looking real images and no external library is employed. In contrast, the earlier methods make use of barcode-like patterns or rely on the imaging device to transmit an identification code, which is subsequently used to retrieve the appropriate model from an external database.

Various visual languages are understandable without any computer system. Pictography is the expression of words and ideas through graphical languages that omit unnecessary details, e.g. icons, road signs, etc. Visual languages are also common in cartography, where color scales describe spatial information, such as terrain topography or temperature, and non-spatial features are described by legends. Different approaches have been described in the literature for analyzing graphical annotations. For example, mathematical and physical laws can be automatically interpreted by a machine [10-12]. Such visual languages can be visualized using the concept proposed here.

3 In-Place Augmented Reality

In Augmented Reality applications, virtual models are created by artists and stored in a library. At runtime these models are retrieved from the library, registered and rendered onto the scene, appearing to overlay the real world. In our In-Place Augmented Reality approach, model details are encoded in real world images, captured by the AR system, and extracted from the scene in order to be rendered back over the image of the real world.

The process pipeline of an In-Place Augmented Reality system is depicted in Figure 2. It consists of three main stages: *Authoring, Content Retrieval,* and *Application Use.* In an offline authoring stage, representations of virtual models and their behaviors are encoded in an image and printed on real paper (see Figure 1(a)). Next, in a one-time step the AR content is acquired and extracted from the printed image. Finally, while the application is running, for each image frame the virtual models that were extracted are registered and rendered over the real page. The final augmented result is shown in Figure 1(b). In the rest of this section, we describe each of these steps in detail.

3.1 Embedding Models and Behaviors

In-Place Augmented Reality deals with embedding models and their behaviors in images to allow their extraction when captured by the AR system. A model consists of geometry and texture, while behavior refers to the way the model acts and is displayed in real time. The models and their behaviors are embedded as *objects* and *annotations*. An object is a planar representation of a model, consisting of images representing geometry and texture. Model behavior is represented by annotations. For example, an animation path for an object is represented as a line drawn on the real image.

To make objects and annotations robustly extractable we give objects a predefined color background and paint annotations in a predefined color. To associate an annotation with an object, we place it next to the object. Figure 1(a) illustrates our model representation using a Saint Moritz map. The background defines the terrain's texture and an elevation map specifies the geometry of the terrain. The elevation map is scaled down and placed as a legend in the top right corner. Two skiers and a bus are embedded as icons on a background and the paths annotate their animated routes. The result of the embedding step is shown in Figure 1(b). When captured by the system the icons for the bus and skiers are animated following the paths on the terrain model. Note that in this simple scheme the exact predefined colors used for marking objects and annotations cannot be used in the rest of the image.

Annotations can also be used for defining user interaction. In Figure 3, a boat object is annotated with a hand, which implies the user can move the boat in runtime using keyboard input. Modeling cues modifying geometry can also be assigned by annotations. For example, the sea model, identified by its homogeneous color, is annotated by a wave modeling cue and at runtime a wave effect is applied to it. Annotations can also be used to define physical properties for objects. The skier can slide down the mountain by adding a gravitation annotation (an arrow pointing down) instead of the animation path.

The types of behaviors that are currently available in our system are path animation, user interaction, modeling cues and physical properties like gravitation. We currently support embedding 2.5D models using elevation maps. As noted above, various existing visual languages can be incorporated into the proposed scheme.

3.2 Model Acquisition

At runtime the embedded image is captured by the camera. We first rectify the embedded image using the homography estimated from the vertices of a surrounding black frame, using the method described in [13]. Figure 4(a) shows the captured image and figure 4(b) shows the rectified version of the image.

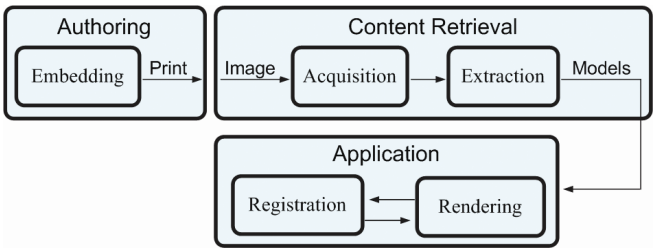


Figure 2. The In-Place Augmented Reality pipeline.

To overcome lighting differences and reduce the effect of camera quality, we next apply image enhancement techniques. There will be blurring caused by the low resolution of the imaging device. To reduce this effect we perform temporal averaging by keeping a short history of the rectified frames. We use a simple scheme to compensate for color distortion. We assume the intensity of each RGB color channel drops independently and linearly with respect to the corners of the image. We add RGB reference color squares to each corner of the image and interpolate each pixel's color according to the brightness reduction from these references. We measure channel intensity reduction in each color square and the intensity increase at black border with respect to the original colors. Equation 1 is used to calculate the corrected value of the *c* channel in the pixel p_i, according to the *r-th* corner.

$$\delta_i^{r,c} = (p_i^c - b_r^c)\frac{1}{v_r^c}, \quad (1)$$

where p_i^c is the value of the *c* channel in the pixel p_i, b_r^c is the *c* channel value measured on the black frame near the *r-th* corner, and v_r^c is the value of the *c* channel measured in the corresponding color reference square of the *r-th* corner. The color value range of each channel is [0, 1]. To propagate this information to each pixel in the image we pre-calculate for each pixel p_i a weighted average according to its distances from the corners. Equation 2 is used to update the value of each pixel.

$$P_i^c = \frac{\sum_r d_i^r \delta_i^{r,c}}{\sum_r d_i^r}, \quad (2)$$

where d_i^r is the distance from p_i to the *r-th* corner.

The result of the acquisition step is shown in Figure 4(c). The image is rectified and has roughly uniform brightness. The virtual model details can then be extracted from the enhanced image as described in the next section.

3.3 Extraction of Models and Behaviors

In the extraction step we reverse the procedure performed in the embedding step. The enhanced image is analyzed and objects with their annotations are interpreted to yield a collection of virtual models with behaviors. The extraction step involves the following operations: (1) extracting models, (2) extracting behaviors, (3) assigning behaviors to models, and (4) generating a background.

Objects and annotations are extracted based on their identifying colors. We perform binarization to keep the appropriate parts of the image by employing three thresholds for the three color-channels. Connected component analysis is then performed by recovering lost pixels using dilation, flooding, and labeling. We drop components that have a small area. Finally, we generate masks for objects and create flat models with the retrieved images as textures.

(a) (b)

Figure 3. Interactive ocean artwork. (a) The boat is annotated for user-interaction using a hand symbol and the sea is annotated by a wave modeling cue using a wave symbol. (b) The augmented result. Note the wave effect and the location of the boat.

In the next step, each of the annotations is identified using normalized cross-correlation template matching against a library of known annotations. The predefined behaviors they represent are then assigned to the nearest models. For example, when a hand annotation is identified, the system wires the keyboard arrows to the model. At runtime, pressing the keyboard arrows will change the object's location. Due to the size of the image, we perform the cross correlation on the discrete Fourier transform, which gives fast and robust identification of annotations. To extract the paths, we perform edge linking followed by thinning [14] on parts that remained unmatched. This yields thin and continuous curves even in cases where curves are slightly broken.

Finally, we create the virtual terrain model from the elevation map and the base image. The elevation map is stretched to fit the size of the base image, assigning height to each pixel, and the base image is corrected and used as a texture. Parts of the image occluded by annotations, objects, and color squares are filled-in using the in-painting method described in [15]. We in-paint the original image, rather than the color balanced one, which gives a rough approximation of the lighting conditions in the real scene.

3.4 Registration

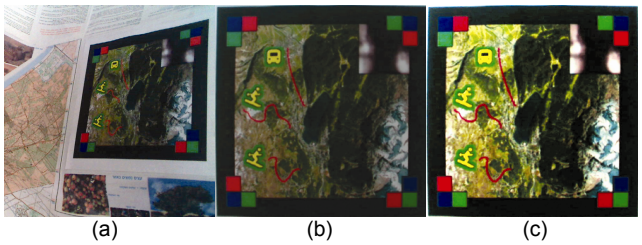
We implemented our system using the ARToolKit registration package [2]. ARToolKit traditionally uses the pattern inside the black frame to orient the marker in space. To consistently determine the orientation of the frame in our system we use a distinct arrangement of colored squares in the corners of the black frame to distinguish each of the corners.

3.5 Rendering

In the rendering stage, the extracted models are rendered onto the scene and modified according to the behaviors assigned to them. Modeling cues are first applied to the geometry at each frame. The effect of user actions is then applied to the appropriate models. Next, animations are calculated and updated. For example, the new position of a path-annotated object is calculated at each frame and the object is transformed accordingly.

4 Implementation and Performance

We implemented an In-Place Augmented Reality platform to demonstrate the feasibly of the concept, test performance, and develop several prototype applications. This was programmed in C++, using the Intel OpenCV [16] image processing library, OpenGL for rendering, ODE [17] for simulating physical properties, and the ARToolKit library [2] for registration. We tested the system using several datasets with good results. The hardware consisted of a laptop with a 1.83GHz Intel dual core processor, 1GB of RAM and a 128MB ATI graphics card. We tested the prototypes with two cameras: a CMOS 960x720 pixel,



(a) (b) (c)

Figure 4. Acquisition of the Saint Moritz map. (a) The embedded image, captured by the camera. (b) The rectified image. (c) The rectified image is averaged and colors are corrected.

30 fps webcam and a CCD 640x480 pixel, 15 fps webcam. The embedded representation was printed on regular A4 paper, using a 600dpi laser color printer.

Content retrieval is performed once and introduces a one-time delay. Image acquisition starts by warping a 960x720 pixel or 640x480 pixel source image to a predefined 300x300 rectified image using the four outer corners of the black frame. We found that the two dominant factors in performance are the rectified image size and the number of objects and annotations. In other words, performance drops as the size of the rectified image and the number of objects and annotations increase. For example, while extracting a single model and its annotations from a 200X200 rectified image takes 265ms, extracting 10 models and their annotations from a 300X300 rectified image takes 921ms. However, in all our experiments the one-time delay of the content retrieval step was negligible.

5 Applications

We have demonstrated the concept of In-Place Augmented Reality in the Saint Moritz ski site map and the interactive ocean artwork applications. Using the ski site map application, we compared the IPAR concept with marker-based AR for people not familiar with AR. We presented it next to a marker-based AR application, which retrieves a digital copy of the same map from a library. The map demonstrates the ski and bus routes on a 3D virtual model of the site, and allows users to understand the topology of the routes. Users preferred the In-Place application since they felt their interaction with it was more natural, because the printed map reflected what they could see in the AR view. The interactive ocean artwork application was used to evaluate different sets of annotations and extraction schemes.

In-Place Augmented Reality applications can be useful in a wide range of different domains. In education, for example, physics books can include printed simulations of physical scenarios demonstrating the studied material. Advertisers creating content for IPAR applications may enjoy a high level of flexibility and simplicity in content distribution and maintenance. Gamers should be able to author their own versions of games and enjoy interactivity in a completely new dimension.

6 Conclusion and Future Work

In this paper we have introduced the idea of In-Place Augmented Reality, which is vision-based transmission of AR content independent of a content library. The idea is based on embedding models into images and extracting them in runtime, where the embedded image is intuitive and understandable to a human without using an AR system. Models are embedded in the world itself, so no database is required to store the models. Thus, the amount of models that can be augmented does not have direct implications on the design and maintenance of the system. Furthermore, the number of concurrent users does not affect the AR system and is hence unlimited. The approach is also useful when no network communication is available. In addition, since AR virtual content is encoded in the real image that it is supposed to augment, separate distribution of content is avoided.

A natural platform for In-Place Augmented Reality is handheld devices. We intend to extend our implementation to such devices and create a flexible framework. We are also currently investigating methods for representing 3D models in printed images. Finally, although we have conducted pilot tests with our prototype, formal user evaluation still remains to be done.

A uniform In-Place Augmented Reality framework with a rich visual language running on different platforms can make AR available to many users and content creators. Our work is a first step in what promises to be a fresh and exciting research direction.

Acknowledgments

This work was supported by the Lynn and William Frankel Center for Computer Sciences and the Tuman Fund. In addition, we would like to thank the reviewers for their constructive comments.

References

[1] H. Saarelma, "Printed Codes — Patent Survey," *Graphic Arts in Finland,* vol. 34, pp. 1-11, 2005.

[2] H. Kato, Billinghurst, M., "Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System," in *2nd International Workshop on Augmented Reality, IWAR99*, San Francisco, USA, 1999.

[3] M. Fiala, "ARTag, An Improved Marker System Based on ARToolkit," NRC Institute for Information Technology 2004.

[4] MXRToolKit, http://mxrtoolkit.sourceforge.net/

[5] Studierstube Tracker, http://handheldar.net/stbtracker.php

[6] Midieum Shin, B. S. Kim, and J. Park, "AR Storyboard: An Augmented Reality Based Interactive Storyboard Authoring Tool," in *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05)*, 2005, pp. 198-199.

[7] L. Tsung-Yu, T. Tan-Hsu, and C. Yu-Ling, "2D Barcode and Augmented Reality Supported English Learning System," in *Computer and Information Science* Melbourne, Australia, pages 5-10, 2007.

[8] J. Rekimoto and Y. Ayatsuka, "CyberCode: designing augmented reality environments with visual tags," in *Proceedings of DARE 2000 on Designing augmented reality environments*, 2000.

[9] The Eye of Judgment, http://www.eyeofjudgment.com/

[10] J. LaViola and R. Zeleznik., "MathPad2: A System for the Creation and Exploration of Mathematical Sketches" *ACM Transactions on Graphics,* vol. 23, pp. 432-440, 2004.

[11] R. Davis, "Magic Paper: Sketch-Understanding Research," *Computer,* vol. 40, pp. 34-41, 2007.

[12] J. A. Landay and B. A. Myers, "Sketching Interfaces: Toward More Human Interface Design" *Computer,* vol. 34, pp. 56-64, 2001.

[13] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge, UK ; Cambridge University Press, 2003.

[14] L. Lam, S. W. Lee, and C. Y. Suen, "Thinning Methodologies-A Comprehensive Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 14, p. 879, September 1992 1992.

[15] A. Telea, "An image inpainting technique based on the fast marching method," *J. Graphics Tools,* vol. 9, pp. 25-36, 2004.

[16] Intel OpenCV, http://opencvlibrary.sourceforge.net/

[17] Open Dynamics Engine, http://www.ode.org/